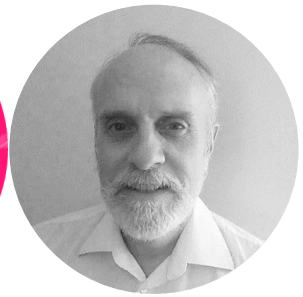
The Observer Pattern



Gerald Britton

Pluralsight Author

@GeraldBritton www.linkedin.com/in/geraldbritton

Overview

Classification: Behavioral

One-to-many relationship between a set of objects

When the state of one changes its dependents are notified

Also known as

- Dependents pattern
- Publish-subscribe pattern

Demo

Dashboard for a tech support center

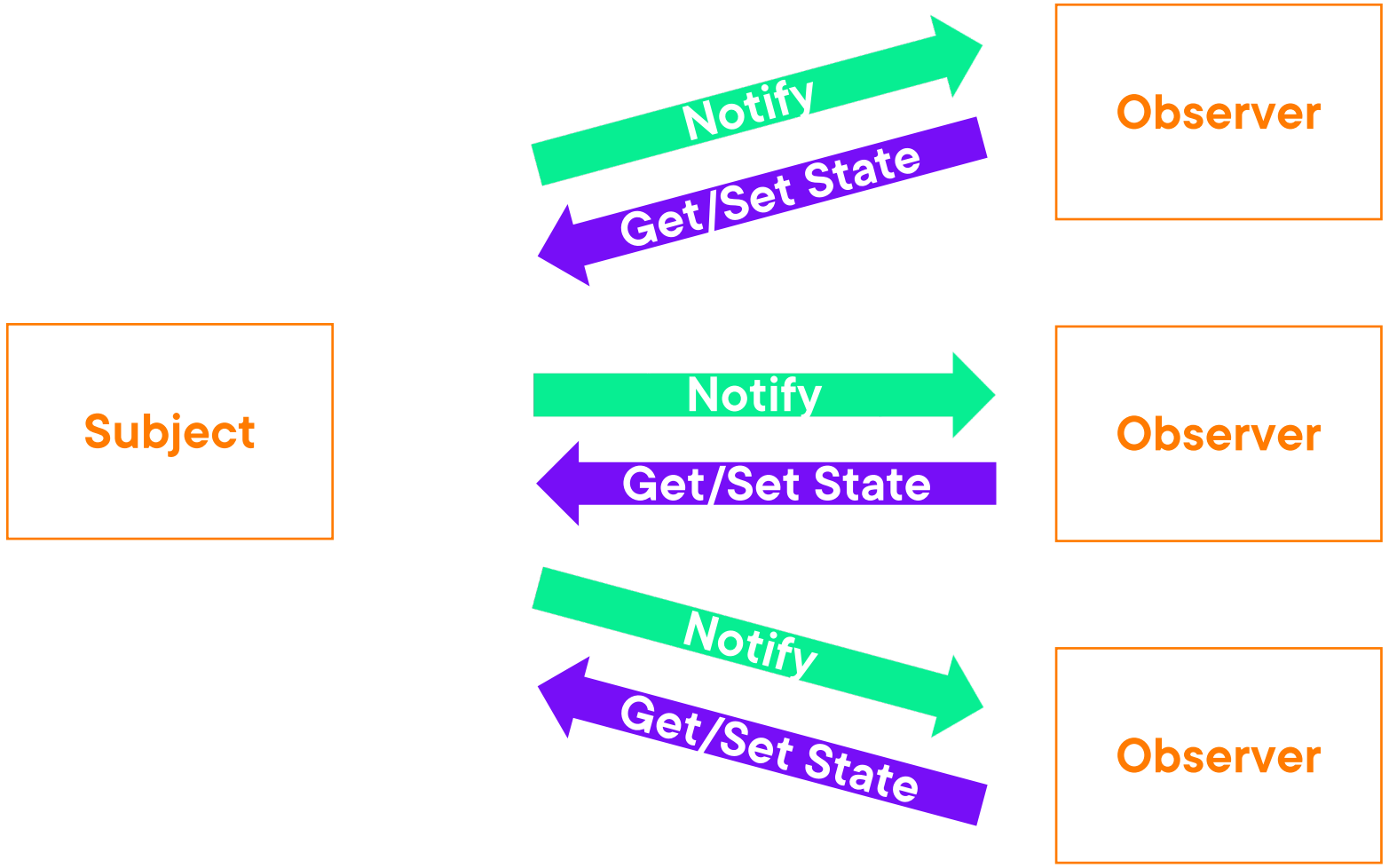
KPIs:

- Open tickets
- New tickets in last hour
- Closed tickets in last hour

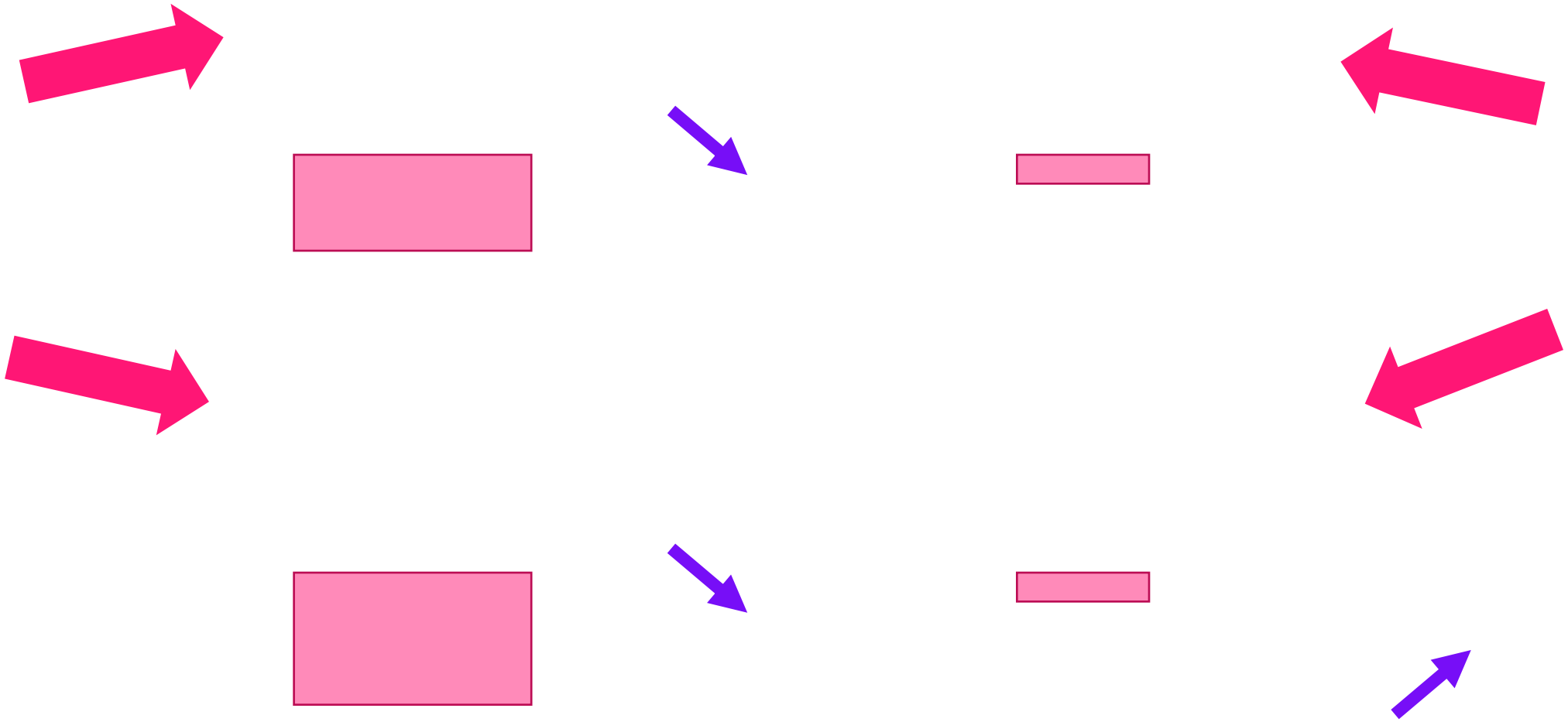
Dashboard is the observer

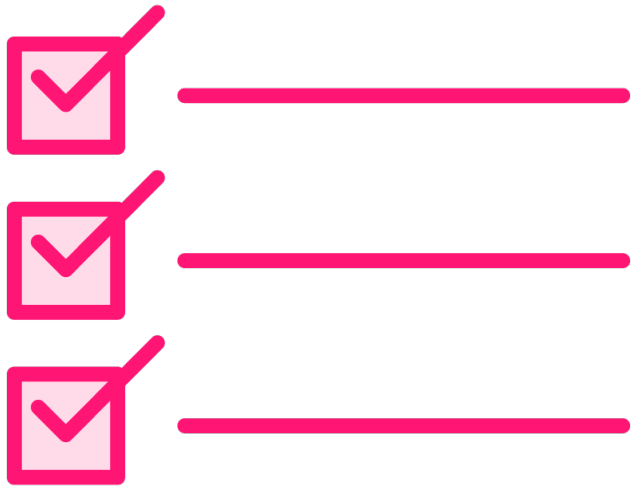
KPI source is the subject or publisher

Observer Pattern



Observer Pattern UML





- Separation of concerns
- Single responsibility principle
- Interface segregation principle
- Open/Closed principle
- Dependency inversion principle
- Encapsulate what varies

Demo

- Implement the classic pattern
- Use ABCs for subject and observer
- Build concrete classes using the ABCs
- Rebuild the main program
- Use two observers

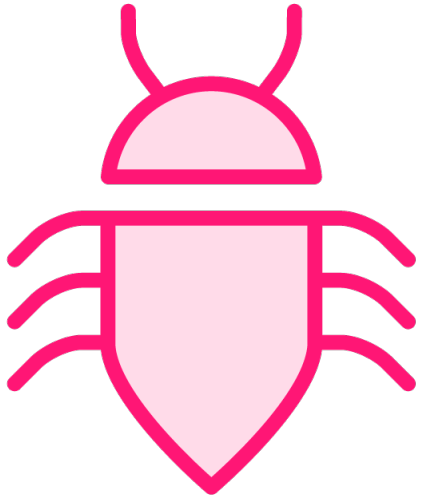
What's Been Achieved?

Implemented the observer pattern

Separated the concerns of subject and observer

Easy to add new observers

One subtle bug



Python runs managed code

Uses reference counters for objects

Set of observers holds references

Need to detach each observer

Why?

If not detached, reference count > 0

Stops garbage collection

Dangling reference

Demo

Use a Python context manager

Change the main program to use “with”

Observers will detach themselves

Subjects will clean up observers

No more dangling references!

Summary

Define a one-to-many relationship

Notify the many when the one changes

Many applications, especially GUIs

MVC pattern:

- Model = Subject, View = Observer

One more thing:

- Extra logic in AbsSubject notify method
- Enables push notifications